

Resumen 2 parcial

ISO



Contents

[2 Memoria 3](#_Toc38873761)

[2.1 Memoria virtual 3](#_Toc38873762)

[2.1.1 Paginacion en Mem virtual 3](#_Toc38873763)

[2.1.2 Tamaño de pagina 3](#_Toc38873764)

[2.1.3 TLB 3](#_Toc38873765)

[2.2 Asignacion de marcos 3](#_Toc38873766)

[2.3 Reemplazo de paginas 4](#_Toc38873767)

[2.4 Thrashing 4](#_Toc38873768)

[2.5 Modelo de localidad 4](#_Toc38873769)

[2.5.1 PFF 4](#_Toc38873770)

[2.6 Demon de paginación 5](#_Toc38873771)

[2.7 Memoria compartida 5](#_Toc38873772)

[2.8 Mapeo de archivos en memoria 5](#_Toc38873773)

[2.8.1 COW 5](#_Toc38873774)

[2.9 Area de intercambio 5](#_Toc38873775)

[3 Entrada salida 6](#_Toc38873776)

[3.1 Servicios 7](#_Toc38873777)

[3.1.1 Formas de realizar E/S 7](#_Toc38873778)

[3.2 Estructura de datos 7](#_Toc38873779)

[3.3 Proceso desde el requerimiento hasta el hardware 7](#_Toc38873780)

[3.4 Drivers 7](#_Toc38873781)

[3.5 E/S mapeada en memoria 8](#_Toc38873782)

[3.6 E/S aislada 8](#_Toc38873783)

[3.7 E/S programada 8](#_Toc38873784)

[3.8 E/S por interrupciones 8](#_Toc38873785)

[3.9 DMA 8](#_Toc38873786)

[3.10 Performance 8](#_Toc38873787)

[3.10.1 Como mejorar esto 8](#_Toc38873788)

[4 File System 8](#_Toc38873789)

[4.1 Metas 9](#_Toc38873790)

[4.2 Conceptos 9](#_Toc38873791)

[4.3 El SO debe 9](#_Toc38873792)

[4.4 Tipos de archivos 9](#_Toc38873793)

[4.5 Atributos de archivos 9](#_Toc38873794)

[4.6 Directorios 9](#_Toc38873795)

[4.7 Comparticion de archivos 10](#_Toc38873796)

[4.7.1 Derechos 10](#_Toc38873797)

[4.8 Asignación de espacio a un archivo 10](#_Toc38873798)

[4.8.1 Formas de asignación 10](#_Toc38873799)

[5 Anexos 12](#_Toc38873800)

[5.1 Asignación indexada ejemplo 12](#_Toc38873801)

# Memoria

## Memoria virtual

Como no es necesario que este la totalidad del proceso subido en memoria puesto que hay partes que capaz se ejecutan una sola vez y no son necesarias o cosas por el estilo, se puede dejar en memoria la parte del proceso que se esta ejecutando y sacar las demás, esta parte en ejecucion es el **Working Set** (WS), el hardware es el que detecta cuando una pagina que no esta en el WS y realiza un trap al SO para que se suba y mientras esto se hace el proceso que necesita la pagina en cuestión se bloquea hasta que se termina la operación, cuando esto sucede se avisa por interrupción y el proceso retorna ejecutando de nuevo la instrucción que dio el fallo. Con este método se pueden tener mas paginas en memoria y puede haber procesos mas grandes que la misma memoria. Para esto es necesario que el HW soporte tanto la paginación como la segmentación por demanda, un disco para el área de intercambio y el SO debe poder manejar los movimientos de paginas entre la memoria principal y secundaria

### Paginacion en Mem virtual

Cada proceso tiene una tabla de paginas en la que cada entrada apunta al frame que la contiene en la memoria principal, además tiene dos bits **V** y **M** que indican si esta cargada en memoria y si ésta está modificada, respectivamente. Los fallos de pagina ocurren cuando un proceso hace referencia a una pagina con su bit **V** en 0 (no está cargada en memoria). De esta manera la tabla de paginas debe contener una entrada por cada pagina del proceso, por lo que su tamaño depende del espacio de direcciones del proceso y del tamaño de pagina que se este usando. Como la tabla de paginas puede llegar a tamaños muy grandes se usan tablas multinivel, básicamente la idea es que la tabla tenga referencias a otras tablas iguales o mas chicas.

#### Tabla invertida

Este método es utilizado en en sistemas donde el espacio de direcciones es muy grande (que espacio de direcciones, ni idea) entonces lo que se hace es una sola tabla de paginas para todo el sistema donde hay una entrada para cada marco de la memoria. Entonces el numero de pagina de un proceso se pasa una una función de HASH donde el resultado es el índice en la tabla de paginas que tiene el marco (hay mecanismos de encadenamiento para las colisiones)

### Tamaño de pagina

Si se usan tamaños mas chicos la fragmentación interna será menor pero las tablas de paginas de cada proceso serán mas grandes

Al usar tamaños de pagina grandes, como los discos suelen mover tamaños grandes los movimientos a memoria son mas eficientes

A menor cantidad de bits de desplazamiento el tamaño de marco o pagina será menor y habrá mas marcos

### TLB

Es una cache que contiene las entradas de la tabla de paginas que fueron usadas recientemente, por lo que para cada dirección virtual se busca en la TLB.

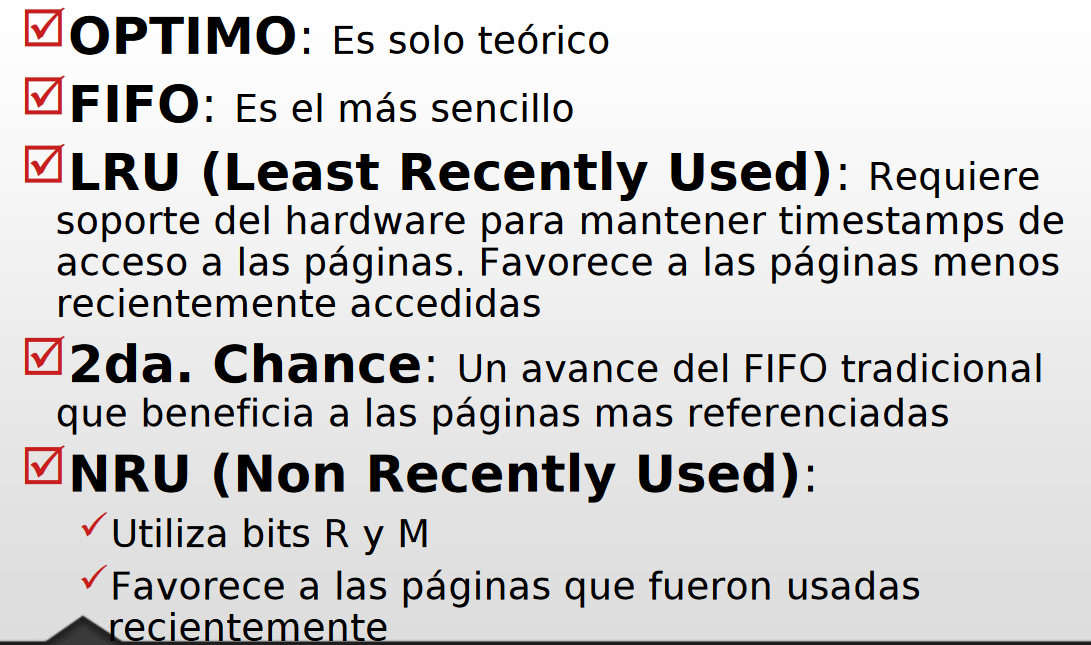
## Asignacion de marcos

Básicamente se pueden asignar marcos de dos formas, fija y dinámica.

En la fija todos los procesos tienen la misma cantidad de marcos asignados y en la dinámica esto varia según cada proceso

## Reemplazo de paginas

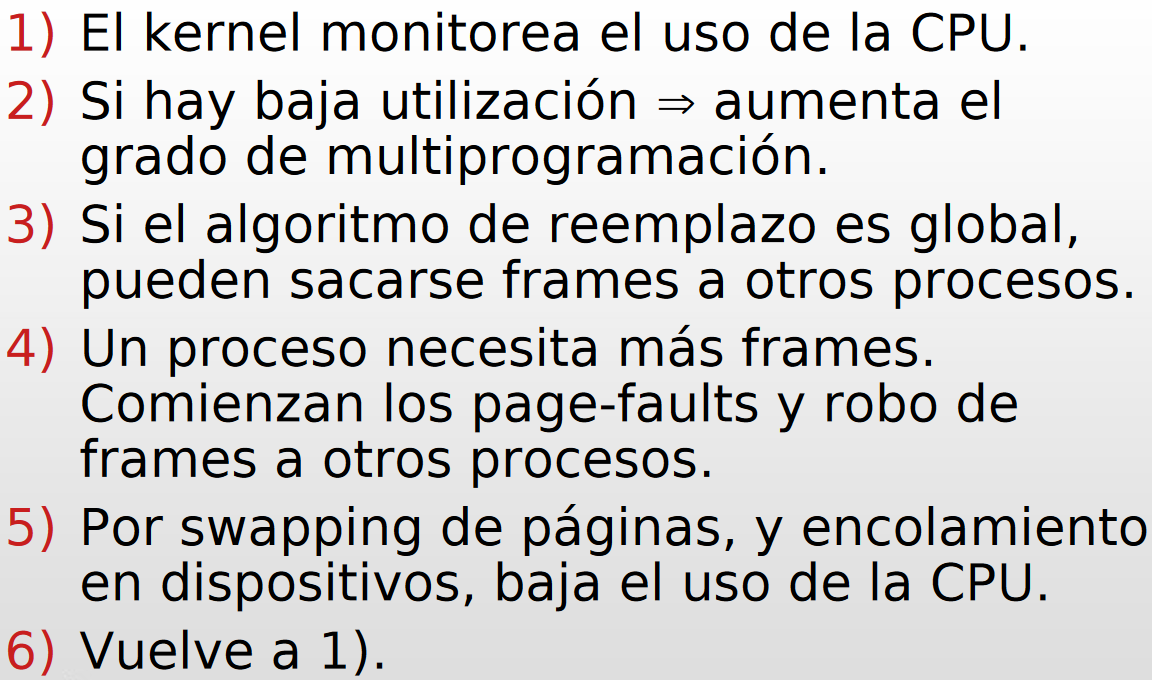
La pagina optima a reemplazar es aquella que nunca mas va a ser referenciada, esto es solo teorico puesto que es imposible de implementar. Todos los algoritmos de reemplazo se basan en las referencias pasadas. El reemplazo puede ser de dos formas, global o local. En el global se pueden elegir paginas de cualquier proceso en el local solo se reemplazan paginas del mismo proceso



OPTIMO, FIFO, LRU, segunda chance, NRU

## Thrashing

Un sistema esta en thrashing si pasa mas tiempo paginando que ejecutando procesos.

Proceso de thrashing

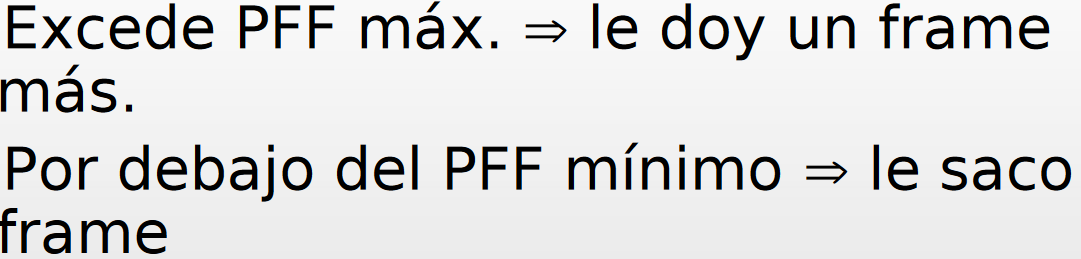
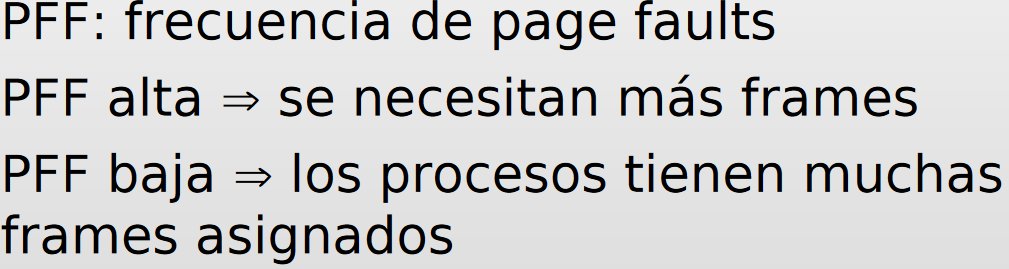
Una forma de controlarlo es con reemplazo local, baja un poco el rendimiento del sistema pero es algo controlable

## Modelo de localidad

La localidad de un proceso es básicamente el conjunto de paginas en memoria en ese momento, mas o menos un programa tiene muchas localidades, como funciones por ejemplo, se podría ver que cada función es una localidad, por lo que se debe elegir un WS que cubra una localidad

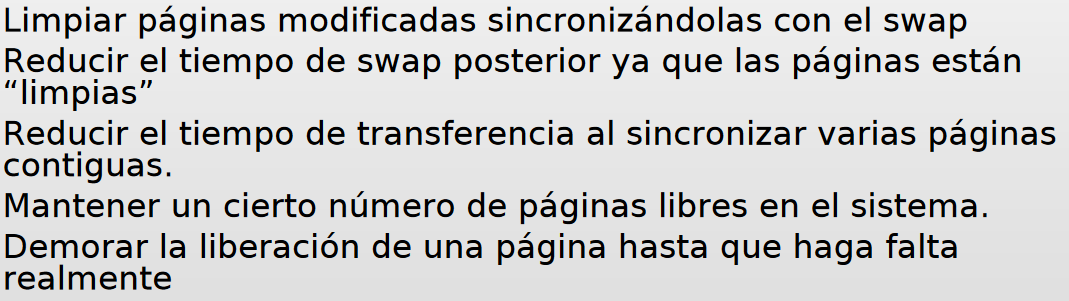
Por lo que una forma de controlar el thrashing es asignando a cada proceso lo que pide, si la demanda total de frames crece excediendo el total de frames suspede procesos y reasigna sus marcos

### PFF



## Demon de paginación

Es un proceso creado por el SO durante el arranque para ayudar a la administración de memoria se activa si hay poca memoria libre o mucha memoria modificada



## Memoria compartida

Varios procesos tienen en su tabla de pagina una pagina asociada a un mismo marco

## Mapeo de archivos en memoria

Se asocia una parte del espacio de direcciones del proceso a un archivo de la memoria secundaria, entonces cuando se haga referencia algunas de las direcciones asociadas con el archivo este se cargara en la memoria, el contenido se sube directo desde el archivo. Cuando el archivo se libera sea por que se desasocian las direcciones o por que termina el proceso directamente, las paginas modificadas se bajan al archivo

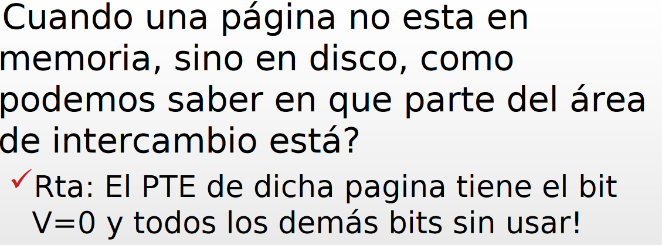
### COW

Cuando se crea un proceso este tendrá su tabla de paginas con punteros a los frame del padre, cuando este modifique alguna pagina, esta se copia y pasa a ser solo del proceso hijo

## Area de intercambio

Si entiendo bien la diapo, hay dos formas de administrar el área de intercambio

1. Cuando se crea un proceso a este se le asigna un espacio en el área de intercambio igual a su tamaño y este pasa a tener una suerte de puntero a la primera dirección en disco de su área de intercambio entonces para acceder a esta se hacer numero de pagina mas la dirección del disco
2. Agarradisimo con pinzas pero entiendo que, cada vez que se saca una pagina de la memoria principal se le asigna un espacio en el disco que se libera cuando la pagina vuelve a memoria, para que esto función, en memoria hay que tener cargado perma una especie de lista de las paginas que están en disco y donde están

exactamente que función cumple esto, se supone que cada vez que se necesita una pagina que se fue, se busca asi? idk

#### en Linux

aparentemente Linux deja tener varias áreas de intercambio, mejor dicho un numero fijo de ellas, a su vez cada área se divide en un numero fijo de slots para paginas según el tamaño de pagina, cuando una pagina se lleva al disco Linux guarda en el PTE el numero de área y un desplazamiento (para el desplazamiento hay 24 bits por lo que el tamaño del área se limita a 64 GB) por que 64 GB si 224 no son 64gb ni a palos pero bueno

# Entrada salida

La idea es manejar todos los dispositivos de E/S de una forma estandarizada encapsulando los detalles de cada dispositivo en las rutinas de los niveles más bajos para que en los niveles altos se vean a los dispositivos como una serie de instrucciones

Algunas formas de clasificar estos son:

* Unidad de transferencia
  + Por bloques
    - Discos
  + Por caracteres
    - Teclado, mouse
* Formas de acceso
  + Secuencial
    - Cintas magnéticas
  + Aleatorio
    - Cualquier otra cosa
* Tipo de acceso
  + Compartido
    - Discos
  + Exclusivo
    - Impresora
* Otro tipo de acceso
  + Solo lectura
    - CDROM
  + Solo escritura
    - Monitores
  + Las dos
    - Los discos
* Legible para el usuario
  + Comunicación con el usuario: impresoras, pantallas, etc
* Legibles para la maquina
  + Comunicación con componentes: discos, cintas, sensores
* Comunicación
  + Interface de red, modem

## Servicios

* Planificacion
* Buffering
  + Almacenar en memoria mientras se transfiere
* Caching
  + Mantener en memoria copia de datos recién accedidos
* Spooling
  + Administrar accesos a dispositivos exclusivos
* Reserva de dispositivos
* Manejo de errores

### Formas de realizar E/S

A priori, dos formas, **bloqueante**, el proceso se bloquea mientras espera a que termine la E/Sy **no bloqueante** cuando el dispositivo termina avisa

## Estructura de datos

El **KERNEL** mantiene la información de estado de cada dispositivo, conexiones de red, archivos abiertos, etc

## Proceso desde el requerimiento hasta el hardware

**Lectura de un archivo en un disco**

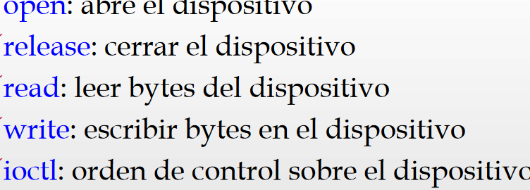
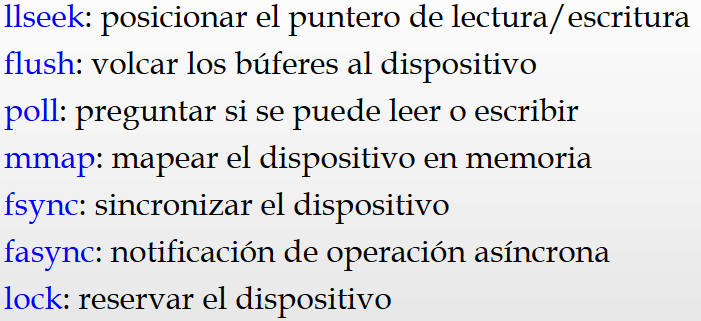
1. Determinar en qué dispositivo esta
2. Traducir el requerimiento abstracto en bloques de disco
3. Leer en memoria secundaria
4. Desbloquear el proceso
5. Retornar el control al proceso

## Drivers

Tienen el código dependiente del dispositivo, manejan un tipo de dispositivos concretos, se encargan de traducir los requerimientos que hacen las rutinas de alto nivel en instrucciones para el dispositivo en cuestión, forman parte del espacio de memoria del kernel y los fabricantes del HW hacen los drivers con APIs del sistema operativo

El SO se comunica con los drivers escribiendo o leyendo de registro que se encuentran dentro de los dispositivos

#### Operaciones

por convención los nombres de las operaciones empiezan con el nombre del dispositivo

## E/S mapeada en memoria

Los dispositivos y la memoria comparte el mismo espacio de direcciones por lo que hacer operaciones de E/S es como escribir o leer en la memoria por lo que no hay instrucciones especiales de E/S

## E/S aislada

Espacios de memoria separados, se necesitan instrucciones especiales y puertos

## E/S programada

La cpu controla toda la operación, claramente desperdicia una barbaridad de cpu puesto que esta debe esperar a que la E/S termine. En este método se hace polling del dispositivo para controlar su estado

## E/S por interrupciones

Los drivers avisan por interrupciones a la cpu cuando el dispositivo termino la tarea

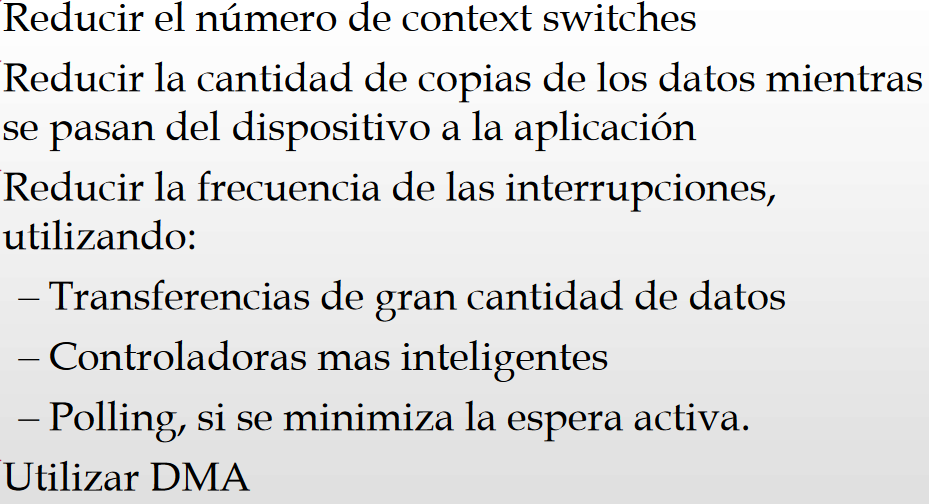
## DMA

El DMA es un dispositivo que controla la transferencia de información entre la memoria y el dispositivo, cuando termina la operación avisa

## Performance

Las operaciones de E/S son claramente la causa que más afecta la performance del sistema en general porque los dispositivos son muy lentos en comparación de la cpu, necesitan mucho tiempo de cpu para los drivers, prova cambios de contexto al hacer interrupciones y bloquear procesos, hacen gran uso del bus de datos

### Como mejorar esto



# File System

Un archivo es una entidad abstracta con nombre y espacio continuo y direccionable

El “File System” es software que provee herramientas para usar los archivos por lo que facilita el acceso a los archivos de las aplicaciones haciendo que los programadores se abstraigan del acceso como tal a los archivos

## Metas

* Dar espacio en disco a los archivos de usuario y sistema
* Mantener un registro del espacio libre, cantidad y ubicación

## Conceptos

* Sector
  + Unidad de almacenamiento y transferencia de los discos
* Bloque
  + Conjunto de sectores consecutivos
* File System
  + Define como son guardados los datos
* FAT
  + Tabla con información de donde esta cada archivo del sistema

## El SO debe

* Gestionar los datos
* Cumplir con las solicitudes del usuario
* Garantizar la integridad del contenido de los archivos
* Dar herramientas de E/S para manejo de archivos

## Tipos de archivos

* Regulares
  + Binarios
    - ejecutables
  + Texto
* Directorios

## Atributos de archivos

* Nombre
* ID
* Tipo
* Localización
* Tamaño
* Seguridad

## Directorios

Es un archivo que tiene información sobre archivos y directorios que están dentro de él, los archivos se encuentran siguiendo una ruta o path compuesta por directorios, por lo que dos archivos pueden tener el mismo nombre pero su ruta completa o fullpathname será única. Al directorio actual se lo llama **Working directory** y dentro de este se puede referenciar a cualquier archivo ya sea por medio del fullpathname o un path relativo

## Compartición de archivos

En ambientes multiusuario se necesita que entre usuarios se puedan compartir archivos, para esto es necesario manejar los accesos (seguridad), siendo el propietario del archivo capaz de autorizar o prohibir el acceso a este a otro usuarios

### Derechos

* Execution
* Reading
* Appending
  + Agregar pero no borrar ni modificar
* Updating
  + Modificar borrar y agregar datos
* Changing protection
  + Modificar los derechos
* Deletion

## Asignación de espacio a un archivo

A priori hay dos formas de realizar esta asignación, antes de que se cargue el archivo y a medida que este necesite espacio.

La pre-asignación es asignar el espacio antes, lo bueno que tiene es que se pueden usar espacios de memoria continuos o lo que mejorara los tiempos de lectura y escritura, sin embargo, cuando te quedas sin espacio ¿?.

La asignación dinámica es ir asignando espacio a medida que se necesita, por lo que los espacios de memoria en disco probablemente no sean continuos

### Formas de asignación

#### Asignación continua

Es básicamente la pre-asignación, por lo que necesita bloques continuos de memoria, tiene fragmentación externa y diría que interna también pero no lo dice a costa de leer en una sola operación

#### Asignación encadenada

Cada bloque tiene un puntero al próximo por lo que no hay fragmentación externa ni interna (esta última quizá si en el último bloque), los archivos pueden crecer y es útil para acceso secuencial aunque random no (ya que es imposible acceder al bloque x de una). En la FAT se guarda el nombre del archivo, el primer bloque y la longitud

#### Asignación indexada

La FAT tiene una única entrada con el bloque de índices o **i-node**, esta técnica tiene distintas variantes como: indexada con secciones e indexada con niveles de indirección.

No produce fragmentación externa.

##### **Indexada con secciones**

No se guardan bloques individuales si no secciones de algunos bloques, por lo que el i-node guarda el bloque inicial y la longitud de la sección.

##### **Niveles de indirección**

**ejemplo**

Cada I-NODO contiene 9 direcciones a los bloques de datos, organizadas de la siguiente manera:

• 7 de direccionamiento directo.

• 1 de direccionamiento indirecto simple

• 1 de direccionamiento indirecto doble

Si cada bloque es de 1KB y cada dirección usada para referenciar un bloque es de 32 bits:

¿Cuantas referencias (direcciones) a bloque pueden contener un bloque de disco?

1 KB / 32 bits = 256 direcciones

¿Cuál sería el tamaño máximo de un archivo?

(7 + 256 + 2562) \* 1 KB = 65799 KB = 64,25 MB

###### Gestión del espacio libre

Control sobre cuáles de los bloques de disco están disponibles

Básicamente hay 4 formas

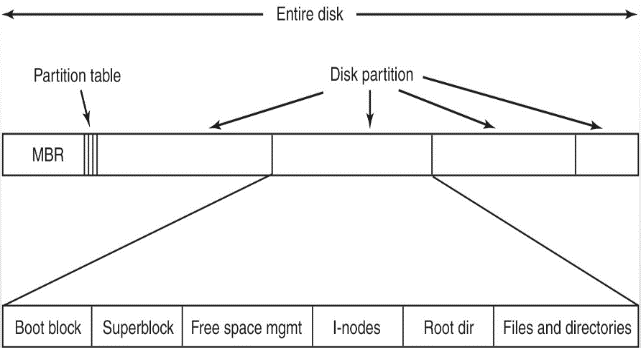
1. Tabla de bits
   1. Un vector de bits, con uno de estos por bloque donde 0 = libre 1 = ocupado
      1. Ventajas
         1. Fácil de encontrar un bloque o grupo de bloques libres
      2. Desventajas
         1. Es pesado por el tamaño que ocupa en memoria
2. Bloques libres encadenados
   1. Cada bloque libre tiene un puntero al próximo
      1. Ventajas
         1. No dicen ninguna
         2. Pero diría que es muy liviano
      2. Desventajas
         1. Hay que hacer un montón de accesos para tener un grupo libre
         2. Se llega a perder un enlace y F
         3. Y encontrar libres consecutivos es casi imposible
3. Indexacion
   1. Parecido al anterior pero mejor, el primer bloque libre tiene N direcciones a bloques libres y la N+1 es una dirección a otro bloque con direcciones
4. Recuento
   1. Es una variante del anterior para mejorar el encontrar grupos libres y consecutivos, la idea es que en lugar de tener N direcciones a bloques libres, se tiene una y la cantidad de bloques libres consecutivos, es útil en asignación continua

## UNIX

### Tipos de archivos

* Archivos comunes
* Directorios
* Archivos especiales (dispositivos)
* Named pipes (comunicación entre procesos)
* Links (comparten el i-nodo solo dentro del file system)
* Links simbólicos (para file systems diferentes)

### Estructura del volumen

* Boot block
  + Codigo para bootear el SO
* Superblock
  + Atributos sobre el file system
* I-node table
  + Tabla con i-nodos
* Data blocks
  + Bloque de datos de los archivos

## Windows

### FS soportados

* Cd-rom dile system CDFS
* Universal disk format UDF
* FAT12 FAT16 FAT32
  + FS originalmente usado por DOS y Windows 9.x
  + Se soporta para compatibilidad
  + El numero indica la cantidad de bits para identificar bloques
    - 212-16-32 \* tamaño de bloque = espacio máximo
    - FAT 32 realmente usa solo 28 puesto que reserva 4
  + Utiliza un mapa de bloques
  + Tiene tantas entradas como bloques
  + La FAT, su duplicado y el directorio raíz se almacenan en los primeros sectores de la partición
  + Se usa **asignación encadenada**
    - Con la diferencia que el puntero al próximo bloque está en la FAT
  + Bloques libres y dañados tienen códigos especiales
* NTFS
  + El FS native de windows
  + Usar 64 bits para referenciar sectores
  + Soporta (comparando contra FAT)
    - Tamaño de disco mayores
    - Mejor performance en discos grandes
    - Nombres más largos
    - Atributos de seguridad

# Cache de disco

Los buffers son la memoria principal para el almacenamiento temporal de bloques tienen el objetivo de minimizar el acceso a disco

Cuando los procesos quieren acceder a un bloque que se encuentra en la cache hay que considerar si el bloque se copia al espacio de direcciones del proceso (esto no permitiría compartirlo) o si se utiliza como memoria compartida (en esta última opción hay que controlarlo con una estrategia de reemplazo) de esta forma los bloques no se mueven si no que se asocian punteros

## Estrategia de reemplazo

Cuando se necesita un buffer para un nuevo bloque se elige el que hace mas tiempo no se referencia (LRU) otra alternativa es LFU

## Buffer cache en System V

El kernel asigna el espacio en la memoria para esta estructura durante la inicialización, cada buffer consta de dos partes:

* Header: tiene información del bloque como nro de bloque, nro de dispositivo, estado, etc
  + Tiene
    - 2 punteros a la hash queue
    - 2 punteros a la free list
    - 1 puntero al bloque de memoria
* El buffer: el lugar donde va el bloque

### Estado de los buffer

Pueden estar disponibles o ocupados (en uso por algún proceso)

#### Delay write DW

Buffers que fueron modificados pero los cambios no se reflejaron en el bloque original

### Hash queues

Son colas para optimizar la búsqueda de un bloque en particular, los header de las colas son un número que es resultado de aplicar una función de hash al número de bloque (dispositivo/bloque) por lo que se busca de la misma manera lo cual es más eficiente

### Free list

Organiza los buffers **disponibles** no necesariamente vacíos, si un proceso uso uno y lo dejo esto puede haber quedado en DW se ordena según LRU. Es la misma idea que la Hash queues pero contiene solo los header de los disponibles (la hash tiene todos). Puede estar en las dos

# Anexos

## Asignación indexada ejemplo

Cada I-NODO contiene 9 direcciones a los bloques de datos, organizadas de la siguiente mariera:

• 7 de direccionamiento directo.

• 1 de direccionamiento indirecto simple

• 1 de direccionamiento indirecto doble

Si cada bloque es de 1KB y cada direccion usada para referenciar un bloque es de 32 bits:

¿Cuantas referencias (direcciones) a bloque pueden contener un bloque de disco?

1 KB / 32 bits = 256 direcciones

¿Cual séria el tamano mâximo de un archivo?

(7 + 256 + 256^2) \* 1 KB = 65799 KB = 64,25 MB